

week 3 design exercise

Tasks

1. Write 250 words that critically reflect on this week's reading:
Malpass, M. 2017. Theories, Methods, and Tactics. Critical Design in Context. London: Bloomsbury, pp.41-70.
2. Create a motion sensor using an arduino, LEDs, and an infrared sensor.

Key Terms

variable

A way of naming and storing a numerical value for later use by the program. The value of a variable can continually change. There are multiple types of variables, including integers, longs, and floats. These are called datatypes. When you call a variable you must declare its datatype.

int

The primary datatype for variables that require numbers without decimals.

long

An extended datatype for variables that require large integers.

float

A datatype for variables that require numbers with decimal points.

camel case

A writing convention for compound words and phrases that does not use spaces and capitalises the first letter of each word, except for the first word. It is used as a naming convention for variables.

Ex: `camelCase` or to create variable for an arduino pin: `int lightShowPin = 3;`

If statement

A statement to test whether certain conditions are met. This statement enables the execution of multiple actions that can be called if the statement is true.

Ex: check if the value of a variable is greater than 30 in order to execute an action.

```
If (someVariable > 30)
{
    doSomething;
}
```

If statements also allow for other types of comparisons:

<code>if (someVariable == 30)...</code>	check if variable is equal to a value
<code>if (someVariable >= 30)...</code>	check if variable is greater than or equal to a value
<code>if (someVariable < 30)...</code>	check if variable is less than or equal to a value

If statements also allow for multiple comparisons

```
if (someVariable > 0 && someVariable < 30)...
```

check if variable is greater than 0 and less than 30.

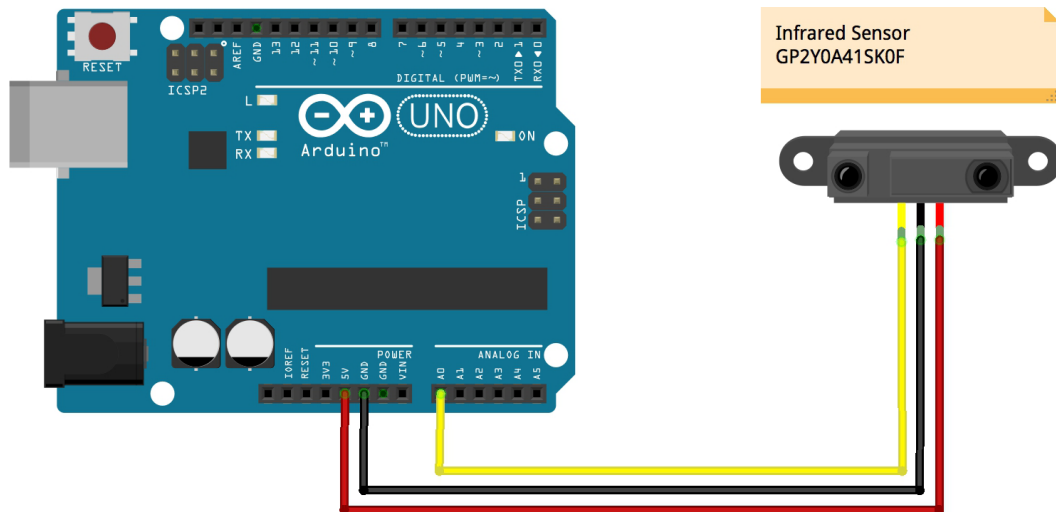
```
if (someVariable == 1 || someVariable == 2)...
```

check if variable is equal to 1 or 2.

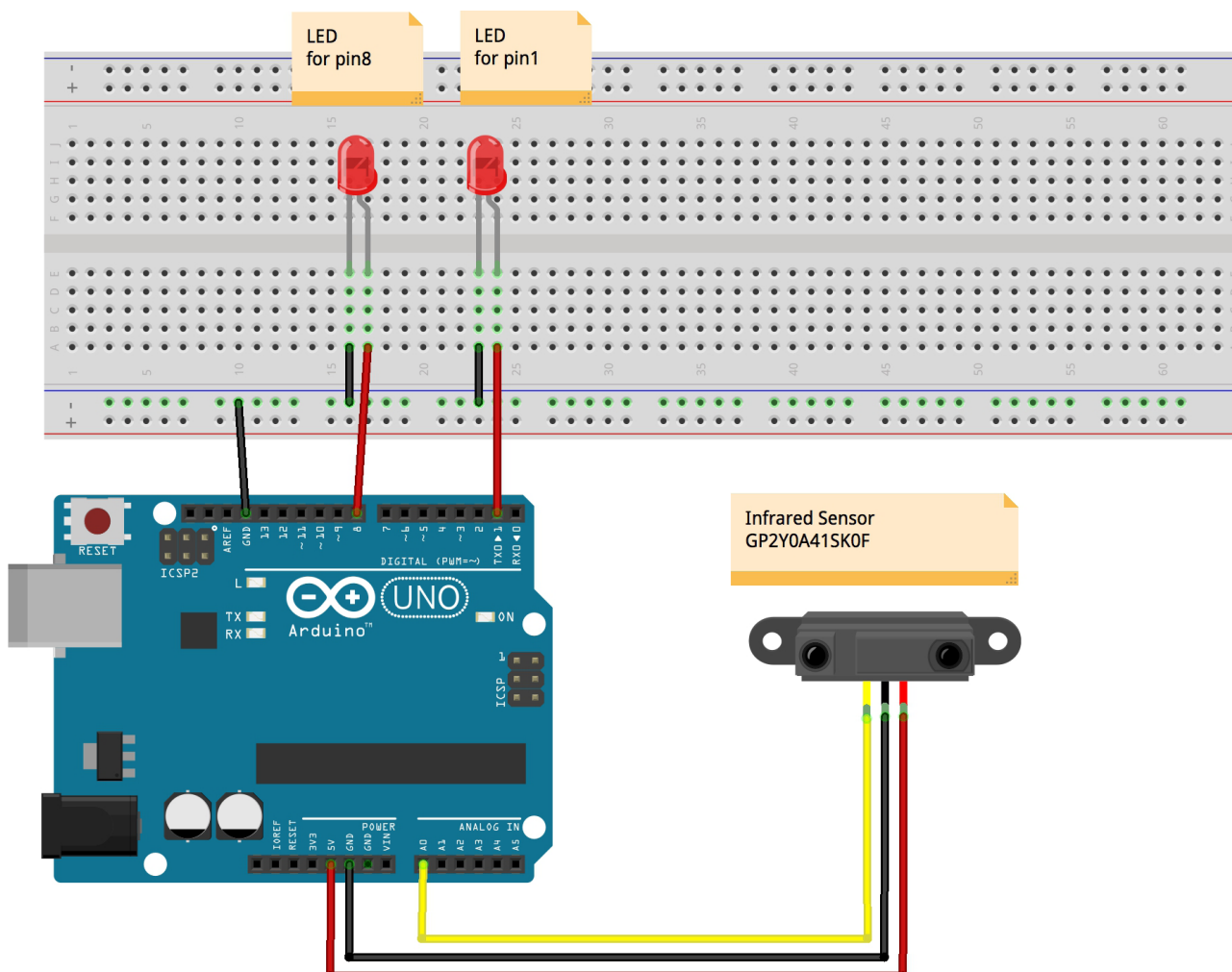
Wiring Setup – Infrared Sensor:

The infrared sensor determines the distance of an object by emitting infrared light from one side and collecting the bouncing light on the other. Using this sensor, create a motion detection system that causes two LEDs to flash on whenever an object passes within 30cm of the sensor.

First, connect the infrared sensor (part no: GP2Y0A41SK0F) to the Arduino using on the diagram below. (note: the colours of the wires in diagram may not be the same as the colours of the wires connected to the infrared sensor.)



Then connect the arduino and sensor to the LEDs using the diagram below. Use a minimum of two LEDs for this assignment.



Programming

Start by programming each component separately: first, the infrared sensor (input); then the LEDs (output); and then combine them together. If you run into any errors, you can easily isolate the issue to figure out what went wrong.

Programming - Infrared Sensor

The infrared sensor requires three variables, which should be written at the top of your sketch.

```
//variables for infrared sensor
int sensor = A0;      /* integer variable assigned to analog pin 0
                        which is connected to the left pin of your infrared sensor */
float volts;          // float variable where sensor input will be translated to voltage
int distance;         // integer variable where to store distance value
```

Then, in the setup function, set initialise the analog pin 0 as an input pin. This will ensure you are reading the sensor value. Then initialise the Serial Monitor, which will allow you to see the sensor values on your computer later on.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialise analog pin for INPUT
  pinMode(sensor, INPUT);
  // initialise serialPort
  Serial.begin(9600);
}
```

Next, in the loop function write the following code to convert the raw voltage from the sensor to a distance value.

```
// the loop function runs over and over again forever
void loop() {

  // get sensor value and multiply to convert to voltage.
  volts = analogRead(sensor)*0.0048828125; // value from sensor * (5/1024)
  // convert volts to distance
  distance = 13*pow(volts, -1);           // worked out from datasheet graph
```

The range of the sensor is between 4 and 30cm, so add the following code to ensure any other values outside of this range will be discarded:

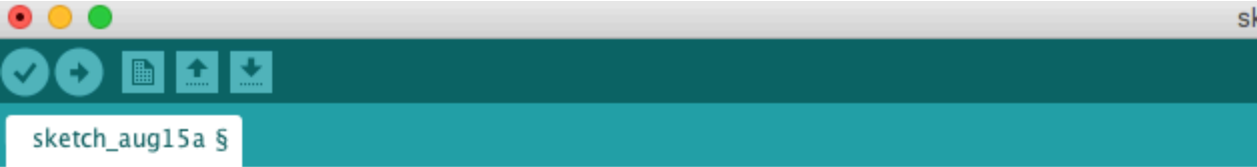
```
// sensor threshold is 30cm, so any value over 30cm will be changed to 0.
// this reduces glitches from the system.
if (distance >= 30) {
  distance = 0;
}
```

Now, still within the loop function, write the following code, which will print the distance value from the sensor to the serial monitor.

```
// print distance value
Serial.print(distance);           // print the distance
Serial.println("cm");            // write cm
```

Programming - Infrared Sensor Continued

Together, the code should look like the following:



```

sketch_aug15a §
//variables for infrared sensor
int sensor = A0;      /* integer variable assigned to analog pin 0
                        which is connected to the left pin of your infrared sensor */

float volts;          // float variable where sensor input will be translated to voltage
int distance;         // integer variable where to store distance value

// the setup function runs once when you press reset or power the board
void setup() {
  // initialise analog pin for INPUT
  pinMode(sensor, INPUT);
  // initialise serialPort
  Serial.begin(9600);
}

// the loop function runs over and over again forever
void loop() {
  // get sensor value and multiply to convert to voltage.
  volts = analogRead(sensor)*0.0048828125; // value from sensor * (5/1024)
  // convert volts to distance
  distance = 13*pow(volts, -1);           // worked out from datasheet graph

  // sensor threshold is 30cm, so any value over 30cm will be changed to 0.
  // this reduces glitches from the system.
  if (distance >= 30) {
    distance = 0;
  }

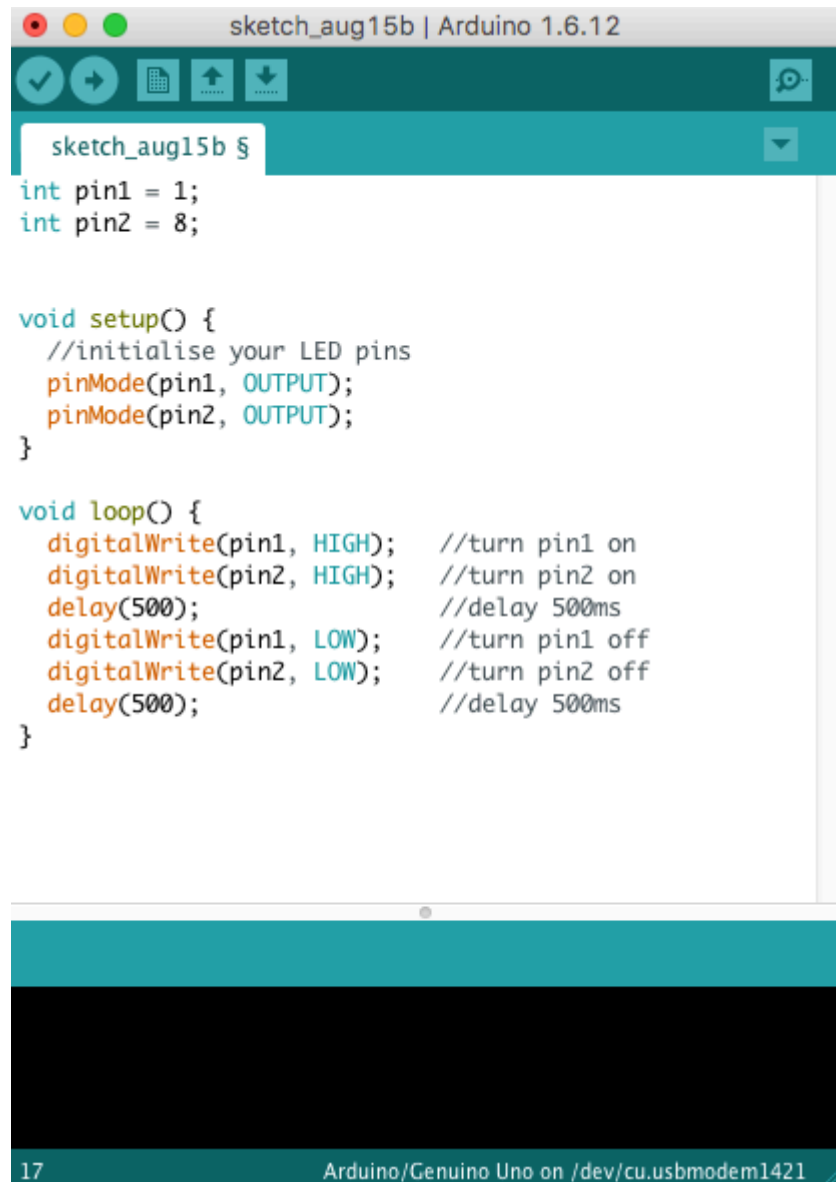
  // print distance value
  Serial.print(distance);                // print the distance
  Serial.println("cm");                 // write cm and return key
}

```

Upload your sketch to your arduino. Once the upload is complete, in the toolbar at the top of the arduino IDE select: Tools > Serial Monitor. The Serial Monitor dialogbox will appear. Distance values from the infrared sensor will be displayed in the Serial Monitor.

Programming - LEDS

Create a new sketch and write the following code, which will turn your LEDS on and off quickly.



```
sketch_aug15b | Arduino 1.6.12
sketch_aug15b $
int pin1 = 1;
int pin2 = 8;

void setup() {
  //initialise your LED pins
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);
}

void loop() {
  digitalWrite(pin1, HIGH); //turn pin1 on
  digitalWrite(pin2, HIGH); //turn pin2 on
  delay(500);               //delay 500ms
  digitalWrite(pin1, LOW);  //turn pin1 off
  digitalWrite(pin2, LOW);  //turn pin2 off
  delay(500);               //delay 500ms
}
```

17 Arduino/Genuino Uno on /dev/cu.usbmodem1421

Upload your sketch to your arduino. Double-check that your LEDS are flashing on for 500ms and off for 500ms.

Programming - Sensor + LEDS

Once the sensor and the LEDs are functioning successfully on their own, it is time to combine them. First, copy your variables from your LED sketch and paste them into your sensor sketch. Then, copy the setup code from your LED sketch and paste it into the setup function of your sensor sketch. It should look like the following.

```
//variables for infrared sensor
int sensor = A0;      /* integer variable assigned to analog pin 0
                        which is connected to the left pin of your infrared sensor */
float volts;          // float variable where sensor input will be translated to voltage
int distance;          // integer variable where to store distance value

//variables for LEDs
int pin1 = 1;
int pin2 = 8;
```

```
void setup() {
  //initialise your LED pins as output pins
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);

  // initialise analog pin for INPUT
  pinMode(sensor, INPUT);
  // initialise serialPort
  Serial.begin(9600);
}
```

Then write an if statement to blink your LEDs if the value of distance greater than 0cm and less than 30cm. Inside the curly braces of the if statement paste your blinking LED code and write something to the Serial Monitor.

```
void loop() {
  // get sensor value and multiply to convert to voltage.
  volts = analogRead(sensor)*0.0048828125; // value from sensor * (5/1024)
  // convert volts to distance
  distance = 13*pow(volts, -1);              // worked out from datasheet graph

  // sensor threshold is 30cm, so any value over 30cm will be changed to 0.
  // this reduces glitches from the system.
  if (distance >= 30) {
    distance = 0;
  }
```

```
// LEDs turn on if sensor is triggered
if (distance > 0 && distance < 30) {
  digitalWrite(pin1, HIGH); //turn pin1 on
  digitalWrite(pin2, HIGH); //turn pin2 on
  delay(500);                //delay 500ms

  digitalWrite(pin1, LOW);   //turn pin1 off
  digitalWrite(pin2, LOW);   //turn pin2 off
  delay(500);                //delay 500ms
  Serial.println("sensor triggered!"); // print "sensor triggered!"
}
```

```
// print distance value
Serial.print(distance);          // print the distance
Serial.println("cm");           // write cm and return key

}
```

Record your video

Record a thirty second video that shows:

1. your hardware connection
2. your final sketch
3. the Serial Monitor Output of your final sketch
4. the functioning systems

This assignment must be submitted in-person at the start of next class to receive credit.
